

Synthesizing 3D Surfaces from Parameterized Strip Charts

Peter I. Robinson (NASA Ames/QSS Group) probinson@mail.arc.nasa.gov

Julian Gomez (NASA Ames/RIACS) jgomez@mail.arc.nasa.gov

Michael Morehouse (Scecor Graphic Arts) mmorehouse@scecor.com

Yuri Gawdiak (NASA Headquarters) ygawdiak@hq.nasa.gov

INTRODUCTION	1
INFORMATION FLOW CHALLENGES IN COMPLEX SYSTEMS	2
FROM STRIP CHARTS TO SURFACES	3
NASA STRIP CHARTS	4
THE TRL SURFACE – FOR NASA PROGRAM MANAGEMENT	4
THE FRAME COUNT SURFACE – FOR ISS CDH FDIR.....	6
DISCUSSION	7
IMPLEMENTATION.....	12
CONCLUSION.....	14
ACKNOWLEDGEMENTS	15
REFERENCES	15

Introduction

We believe 3D information visualization has the power to unlock new levels of productivity in the monitoring and control of complex processes. Our goal is to provide visual methods to allow for rapid human insight into systems consisting of thousands to millions of parameters. We explore this hypothesis in two complex domains: NASA program management and NASA International Space Station (ISS) spacecraft computer operations. We seek to extend a common form of visualization called the strip chart from 2D to 3D. A strip chart can display the time series progression of a parameter and allows for trends and events to be identified. Strip charts can be overlaid when multiple parameters need to be visualized in order to correlate their events. When many parameters are involved, the direct overlaying of strip charts can become confusing and may not fully utilize the graphing area to convey the relationships between the parameters.

We provide a solution to this problem by generating 3D surfaces from parameterized strip charts. The 3D surface utilizes significantly more screen area to illustrate the differences in the parameters and the overlaid strip charts, and it can rapidly be scanned by humans to gain insight. The selection of the third dimension must be a parallel or parameterized homogenous resource in the target domain, defined using a finite, ordered, enumerated type, and not a heterogeneous type. We demonstrate our concepts with examples from the NASA program management domain (assessing the state of many plans) and the computers of the ISS (assessing the state of many computers). We identify 2D strip charts in each domain and show how to construct the corresponding 3D surfaces. The user can navigate the surface, zooming in on regions of interest, setting a mark and drilling down to source documents from which the data points have been derived. We close by discussing design issues, related work, and implementation challenges.

Information Flow Challenges in Complex Systems

Behind complex systems as diverse as NASA program and spacecraft management is a pyramid of information. This information pyramid exists so that humans and machines responsible for determining and controlling the state of complex systems will not be overwhelmed by the numerous lower-level details. On the other hand, these details are important, for without the lower-level information, higher management is more likely to making uninformed decisions. For instance, accident reports from the NIAT[2] and CAIB[3] report cited lack of information flow as a significant factor in the failure of both Space Shuttle and Mars spacecraft. In addition, as mission controllers of ISS and future missions start to rely on fewer people to monitor and control more subsystems (e.g. spacecraft built in stages) [4], a need will grow for tools that provide a systems wide view and as well as the relevant details.

For NASA program management, information rises through a managerial hierarchy, progressively concerned with strategic issues over timescales of quarters and fiscal years, rather than tactical issues occurring over timescales of days to weeks. Similarly, in the NASA spacecraft operation domain, the information concerning the spacecraft operation rises through a hierarchy of data processing, from the onboard computers and astronauts to ground computers and mission controllers to the spacecraft operations head. In the program management domain, a lack of understanding of root causes for meeting and/or slipping milestones might result in incorrect funding decisions. In the spacecraft operation domain, a risk for incorrect spacecraft operation decisions exists when the root causes and ramifications for low-level events are not understood.

Therein lies our challenge, providing decision makers with a high-level view that will not flood them with details, but at the same time, allowing them access to the details in case they need the data. These constraints conflict. Our solution is a derivative of the ancient saying, "A picture is worth a thousand words." In our case, a 3D surface is worth ten thousand data points. Presenting ten thousand words to the user map will swamp the user, but providing a single picture enables the user to scan, for a high-level view, and to search and drill down for relevant changing details

From Strip Charts to Surfaces

Strip charts are a common method for displaying data over time. By plotting data over time, trends and events can be identified. Overlaying strip charts is a way for discovering when events on multiple strip charts correlate. Below are a few examples for displaying 3D strip chart data over time. In Figure 1, Methods 1-4 demonstrate the use of graphing 3D strip chart data. None of the methods connects the strip chart elements to create a single strip chart surface. That's because to construct a single 3D surface from strip charts, first an **order** of the strip charts must be defined.

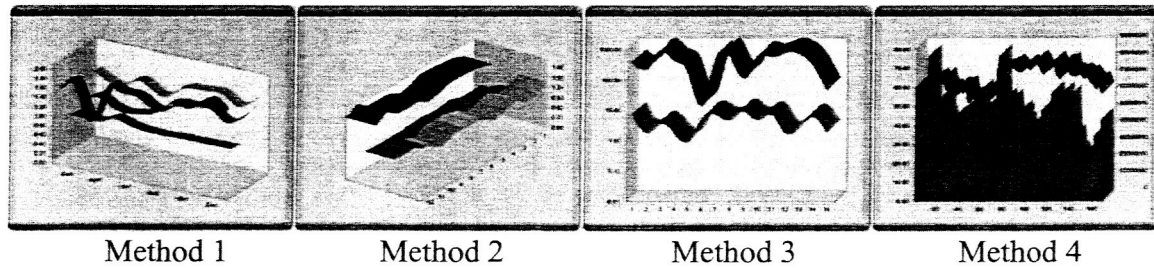


Figure 1 A survey of four 3D methods for visualizing strip chart data.[28]

By introducing an order among strip chart elements, an extruded surface connects values of strip charts at points in time. The third dimension is often a discrete enumerated type over which an ordering relationship can be defined.

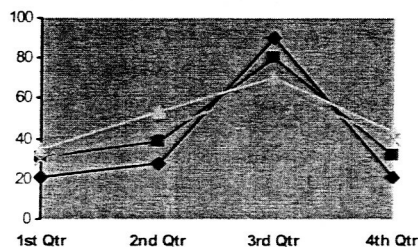


Figure 3 Overlaid Strip charts

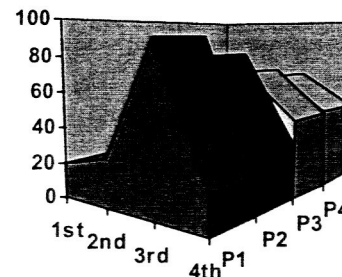


Figure 2 Strip-chart surface.

For example in Figure 3, four representative parameterized strip charts are overlaid in 2D where in the third quarter all of the signals have a maximum. In Figure 2, given a domain-specific method of ordering the same four strip charts are displayed as a set of surfaces, where the third quarter maxima can also be observed. Now the surface can be used to add fourth and fifth dimensions for color and texture. This approach to surface generation is useful if the strip charts are of homogenous type (e.g. set of pressures, set of temperatures, set of plan/milestone Technology Readiness Levels (TRLs)[1], set of processor frame counts). The surface may lose its semantics if the types are heterogeneous (e.g. pressures and temperatures).

By transitioning from an overlaid strip chart 2D visualization to a 3D strip chart surface, we have increased the portion of the display that is related to the content, namely the comparisons of the quantities over time. For example, in Figure 3, where we show the overlaid strip charts, the content/frame ratio (ratio of full plotting area to that used to

display content) is much less than the content/frame ratio for the surface in Figure 2. In addition we increase the data density of the visualizations when we use the surface to represent parameters that are mapped to color and texture.

NASA Strip Charts

We illustrate strip charts for both the NASA program and spacecraft management domains and proceed to develop surfaces from the strip charts.

For program management monitoring and control, parameters measured over time could include funding levels, TRL[1], risk level, and status information. In the NASA program management domain, the TRL levels of a single program plan can be plotted as TRL vs. time over the lifetime of the three-year plan (Figure 4). The TRL parameter is a very important parameter by which the maturity of NASA technology is measured [1]. The change in TRL over time is a strong indicator of whether technology research and development is proceeding according to plan.

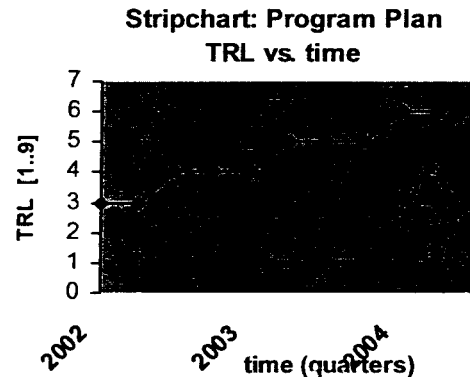


Figure 4 TRL Strip chart <time, TRL>

For spacecraft monitoring and control in the ISS Command and Data Handling (CDH) computer domain, monitored parameters could include the frame count heartbeat from the onboard computers[5], as well as the flow of information through the computer buses and memory. In the NASA spacecraft management domain, we model the frame count of single computer onboard ISS as frame count vs. time (Figure 5), each saw-tooth cycle requiring ten seconds. The frame count parameter is the measure of the "heartbeat" parameter which each ISS computer is required to have for nominal operations. Other parameters are program state and an onboard global shared memory called the Current Value Table (CVT)[9].

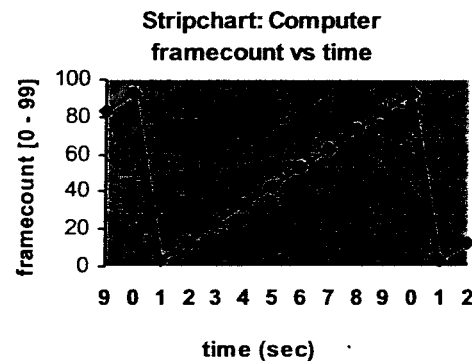


Figure 5 Frame count Strip chart : <time, frame count>

The TRL Surface – For NASA Program Management

For the NASA program management domain, we have selected the **plans** parameter as the third dimension for the construction of the TRL surface. Together with the strip chart axes of **time** and **TRL** (Figure 4), we can define our 3D space by introducing the **plans** axis. We map each 2D data point <time, TRL> to 3D data points of the form <time, plan, TRL> (Figure 7 and Figure 6). The TRL surface is then constructed by defining a grid over the set of 3D data points (see Implementation section). The grid is constructed

from line segments in both the **plans** and **time** dimensions. The line segments in the **time** dimension correspond to the original strip charts which model how a single plan's TRL changes over time (Figure 4). While the line segments in the **plans** dimension model the TRL state of all plans at a point in time. The surface generated can be seen in Figure 6.

Methods for choosing the order of the strip chart's slices along the **plans** dimension rely on domain specific information. For the TRL surface in Figure 6, the order of the plans has been defined by the Level 2 AVsP plan[6]. A Level 2 plan is made up of a set of Level 3 plans such that the Level 3 milestones are "rolled-up" to define Level 2 milestones. The values in the **plans** dimension for the AVsP plan are discrete, finite, and enumerated: 2.1.1, 2.1.2, 2.1.3, and 2.1.4. The rollup relationships in the Level 2 plan provide the dependencies to order the Level 3 plans. The upward slope of the surface over time indicates that the Level 2 plan expects the milestones of the Level 3 plan to increase in TRL as time passes. The color of the surface indicates the amount of funds spent at each point in time. In fact, the slope increases front-to-back through the **time** dimension as well as left-to-right through the **plans** dimension. The front-to-back increase is due to the passage of time. The right-to-left increase is attributed to milestones in a later plan being dependent upon milestones in earlier plans.

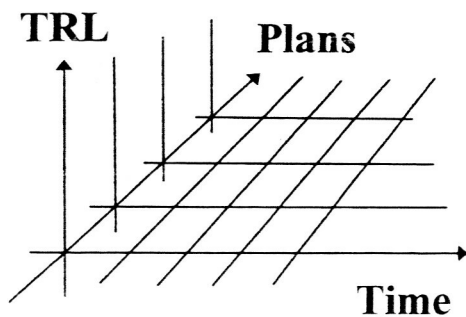


Figure 7 TRL 3D Axes.

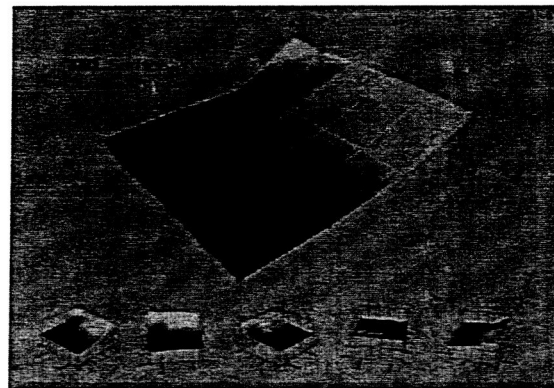


Figure 6 TRL Surface: <time, plan, TRL>

The TRL surfaces provide a way to quickly recognize trends in large data sets by mapping surface height change to changes in plan TRL level, and color saturation change to changes in plan funding level. We believe that by scanning the surface, it will rapidly convey information to the program manager concerning the strategy for maturing and monitoring technology as well as provide, through data dependencies, an entry point to source material.

Many extensions to the TRL surface are required to increase its utility. The TRL surface does not reflect the reality of the facts on the ground. The TRL surface must incorporate realtime information in the form of monthly and quarterly status update information in order that the traditional red, yellow, green status indicators can be compared against plan expected progress. (e.g. [7]). The color and texture of the TRL surface will be used to incorporate this knowledge and will allow comparison of how well the research and development plan was/is able to achieve the actual TRL levels. In addition, once the dependency mappings are constructed between the TRL surface and source documents,

the user can navigate the TRL surface to project plans and access to monthly and quarterly reports when more detailed knowledge is required.

The Frame Count Surface – For ISS CDH FDIR

For the NASA spacecraft domain, we have selected the **processors** parameter as the third dimension for the construction of the frame count surface. Together with the strip chart axes of **time** and **frame count** (Figure 5), we can define our 3D space by the **processors** axis. We map each 2D data point **<time, frame count>**, to 3D data points of the form **<time, processor, frame count>** (Figure 9 and Figure 8). The TRL surface is then constructed by defining a grid over the set of 3D data points. The grid is constructed from line segments in both the **processor** and **time** dimensions. The line segments in the **time** dimension correspond to the original strip charts which model how a single processor's frame count changes over time (Figure 5). While the line segments in the **processors** dimension model the frame count state of all processors at a point in time. The surface generated can be seen in Figure 8.

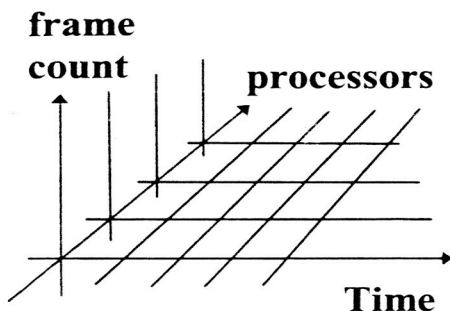


Figure 9 Frame count 3D Axes.

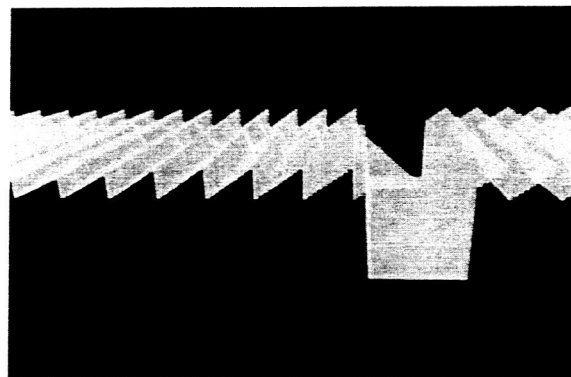


Figure 8 Frame count Surface: **<time, CPU, frame count>**

The saw-tooth nature of the surface over time indicates that the processors are operating in a nominal manner. The change in the sawtooth pattern is caused by the absence of telemetry due to a dropout in the data for all of the processors. The root cause for such a dropout could be a variety of software and hardware components in the information train from onboard processor to ground-based telemetry storage and access[10]. Over 1000 data points are used to represent the surface. We will extend the frame count surface to utilize color to reflect processor status and other information. In addition, by annotating the surface with dependency information the user can navigate to source information related to the state of each processor's operation and design.

Methods to choose the order of the strip charts slices along the **processors** dimension must rely upon information specific to ISS CDH domain. For the frame count surface in Figure 8, the order of the processors has been selected to correspond to the pyramid hierarchy of processors utilized onboard ISS. The hierarchy of the processors identifies the physical dependencies that exist between the computers. We have ordered the processors according to their location in the ISS three-tiered computer hierarchy. Top level computers control the ISS state, mid-tier computers control subsystems, and lower tier computers perform the actual sensing and control of ISS. The source of the data is the

ISS telemetry downlinked to Earth[8] once a second. The values in the **processors** dimension are alpha-numeric parameters known as device process unique identifiers (PUIS) [9]]. For example the frame count PUIs for upper tier CCS_Primary and Backup computers have: alpha-numeric values LADP01MDZZ01U and LADB02MDZZ01U respectively.

Discussion

To ensure effective use of the 3D surfaces we address several criteria: 1) interactivity and navigation – By what principles and how will the user interact with the 3D surfaces in order to gain insight and navigate to proper sources? 2) surface features and definition – How to define relationships between surface features and the quantities of interest? How are they implemented? 3) “escaping flatland” – How to use surfaces to maximize the number of dimensions which can be shown on a flat two dimensional screen. Below we address these three areas.

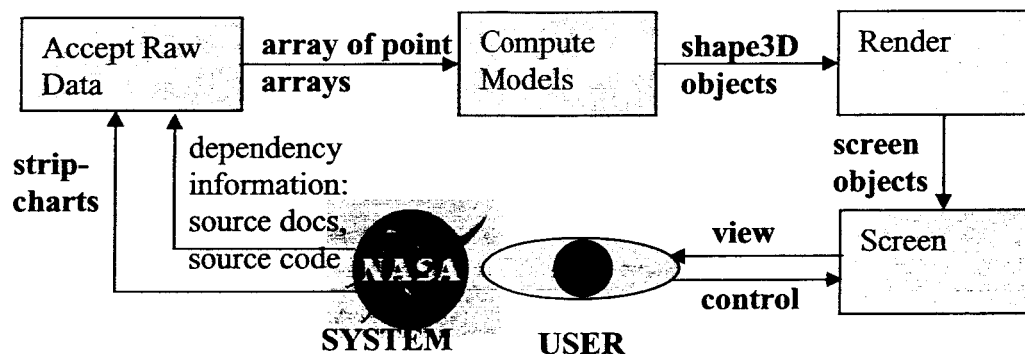


Figure10. Information processing flow of a model-viewer-control (MVC) architecture for interactive 3D surface visualization. System accepts sets of strip charts as well as dependency information and presents 3D views to the user, allows user to drill down to source documents.

Interactivity and Navigation. We employ the classic model-view-control (MVC) methodology (Figure 10). MVC is based on the feedback control principles which underlying modern control theory and AI-based methods. In MVC, the user can iteratively **control** the **view** of a **model**. For example, the user can fly over the surface or fly through the surface to drill down to source documents. In addition, the model and the **data** represented by the model are separate such that both static data as well as the dynamic data (e.g. telemetry) can be viewed. The control of the view allows the user 1) access to the data 2) manipulation of the model objects and 3) navigation of the views [14].

The iterative process of viewing and controlling the 3D visualization must address what Andrew Lippman[16] has defined are the five criteria of interactivity required to fulfill “the give and take of two participants [computer and human]”:

- **interruptibility:** both the human and computer can interrupt the process
- **granularity:** what is the smallest unit of control or viewing which supports interruptibility

- **limited lookahead:** avoid pre-computing of surfaces – instead derive from database of data and source documents especially when visualizing realtime data.
- **graceful degradation:** when the system cannot control in a manner desired by the user, provide methods to check-point back to previous known good state as well as record unsuccessful control requests for analysis by system developers.
- **the appearance of infinitude:** give “the impression of an infinite database” of possible directions for navigation and exploration (e.g. surface of sphere: finite surface, infinite extent).

We implement the interactivity principles in the commands which are available for the system. Tufte suggests that “the number of computer commands immediately available (more the better), if clearly but minimally displayed” [15]. We partition control of the system is into four areas: 1) control of the view 2) set point/mark 3) drilling down to sources and 4) preferences.

- **Control of the view** so the user can navigate the surface to zoom in/out on regions of interest, pan, rotate, scale and fly through (could also export to VRML which has extensive fly through capabilities).
- **Set point/mark** on regions of data selected for analysis. Users will be able to rubber-band regions of the surfaces (Figure 11). Users can select: 1) time slices (show all slices @ time) 2) strip chart slices (show a slice @ all times) and 3) arbitrary regions of time. In the future, the point/marks will be available across all displays of the system to ensure consistency and parsimony of navigation.
- **Drill down** to source documents from which the data points were derived. The surface is a visual metaphor for a set of low level data. The data is dependent upon a vast set of more detailed source material. For example in the TRL domain, the surface represents the Level 2 plan milestones made up of a set of Level 3 milestones plans which are made up of all the product generated by the execution of the plans. While in the frame count surface example, the surface represents the coordinated frame count execution of all the Tier 1 and Tier 2 computers, themselves defining the execution of a vast set of multiple programs executing with thousands of variables. We will provide capabilities which allow navigation between “a broad overview to the fine structure”[15]. Navigation will be accomplished by encoding underlying dependencies between data points and source documents in a domain specific manner. In the implementation section we illustrate how to augment the 3D data structures to support dependency information.
- **Preferences.** User can select colors, axes, orders of strip charts in order to scale, and reorder surfaces as necessary.

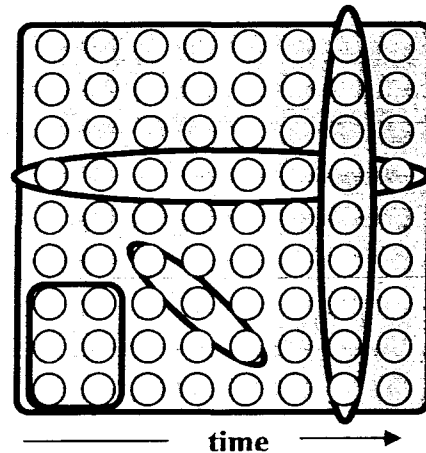


Figure 11 3D Set point/mark also 3d surface points to be selected in a variety of groups.

Surface Features and Abstract Quantities. In “Discovering Visual Metaphors”[11] Gershon and Page state: “Unlike scientific visualization, information visualization

focuses on information that is often abstract, thus lacking natural and obvious physical representation.” We have faced this challenge by limiting our physical representation to that of a colored/textured 3D surface. The dimensions for the 3D colored/textured surface are then defined by developing domain specific mappings between 3D surface features (x, y, z, color, texture) and the abstract quantities of interest (e.g. funding levels, TRL[1], risk level in the program management domain and status information, frame count, program state in the ISS CDH spacecraft domain).

Tufte also provides guidance in defining how to visualize quantities. He states that quantities can be visualized in three different ways: direct labels, encodings and self-representing scales. He provides a guide to their use through a strategy of using the smallest effective difference: make all visual distinctions as subtle as possible, but still clear and effective[17] by:

- Direct labels are defined with respect to the grid axes, the 3D points for each parameter preserving the “relative difference” between the values of quantity, as represented by “relative distances” on the 3D surface.
- Encodings map a quantity to either the color or texture of the surface. When using color we do not use multi-color representations as Tufte cautions against (see Sea of Japan example[17]), instead define a value scale which progresses from light to dark, (e.g from light blue to dark blue) and suggests the use of organic colors from nature.
- Self-representing scales identify portions of the visual display which are duplicated across the visual field. Since each replicated image corresponds to an instance of quantity, the difference in the size of the replicated images allows for comparison of the quantities. On our surfaces, the replicated images we utilize are the individual strip charts themselves. The slope of the 3D surface highlights the change in the shapes of the replicated strip charts and supports comparison of quantities across the replicated parallel images.

In the table below we summarize the quantities in the two NASA domains:

Domain	Direct Labels				Encodings	Self-representing scales
TRL Surface	time	plan	TRL	axis	color	The plan type replicates through the y-axis.
	3 years	[0..maxplan]	[0..9]	time	Cost \$, status	
	1 month	1	1	plan		
Frame Count Surface	time	processor	Frame count	axis	color	The strip chart type replicates through the y-axis
	5 min.	[0..60]	[0..99]	time	ISS trider state[10]	
	1 sec	1	1	processor		

Figure 12 Table of surface features in the TRL surface and frame count surface domains defined using the methods of visualizing quantities of parameters.

“Escaping Flatland”[18]. Tufte observed as many have, that visualizations are trapped in two dimensional “flatland” even though the state vector for most visualization domains is far greater than two dimensions. Over the ages “flatland” has consisted of the walls of the Lascaux Caves, the stone tablet, papyrus, paper and now the computer screen. Menard’s 2D visualization of Napoleon’s Russian Waterloo demonstrates [18] (Figure 13) how to model six dimensions without the use of 3D figures. By incorporating Menard’s lessons together with the use of perspective drawing to model 3D shapes, we can develop 3D metaphors which can model even more than Menard’s six dimensions.

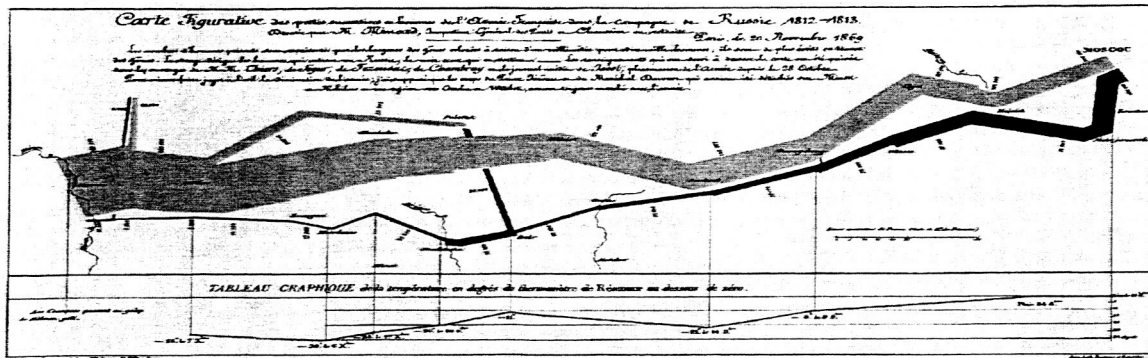


Figure 13. Menard’s 2D Visualization of Napoleon’s 1812 Waterloo – 6 dimensions modeled in “flatland”: 1) line color: direction or army (brown/grey: to Moscow, black: retreat from Moscow), 2) line thickness: quantity of troops, 3,4) line position: (x,y) location on map of Poland), 5,6) temperature @time: vertical drop to temperature on its own scale @time.

Perspective drawing provides the first three dimensions for our surfaces. For example in Figure 14 two 3D surfaces are defined that analyze presidential election year data from 1956 and 1960. The three axes are, one continuous axis and two discrete axes: x) party identification: [Democrat (strong, weak), Independent, Republican (weak, strong)], y) religious identification: [Protestant(strong,weak), Catholic(weak, strong)] , z) democratic percentage of the two-party vote: [0..100]. The 3D surfaces show a marked upward trend in the maxima of all three dimensions and other changes as well on the surface. These

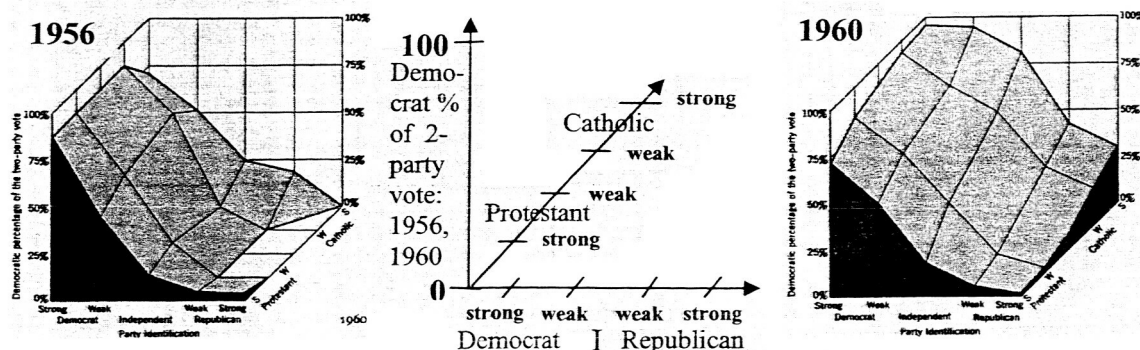


Figure 14 3D Surfaces Representing Comparison of Voting Patterns between 1956 and 1960 [27].

two examples through changes in the shape of the surface, allows the viewer to grasp the effect that candidate John F. Kennedy, who became the United States’ first Catholic president, had on voter participation. Comparing the two surfaces is easy, comfortable, and rapid.

As we stated earlier perspective drawing provides the first three dimensions. The fourth dimension is defined by surface color (e.g. light to dark blue) while surface texture (e.g. rough to smooth) provides the fifth dimension. Sixth and higher dimensions are represented by the addition of landmarks and terrain features on the landscape surface.

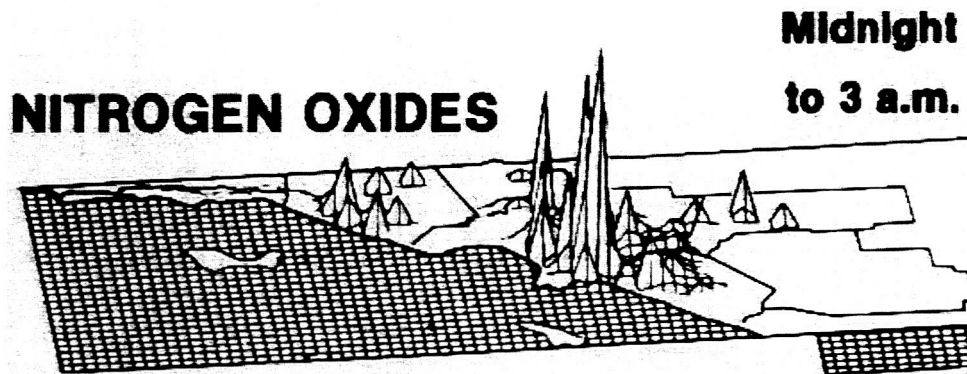


Figure 15 3D pollution data overlayed on map of from Santa Barbara to greater Los Angeles [26]

The use of features on landscapes can be seen Figure 15, a map showing Southern California as the foundation surface for visualization of pollution data. Peaks correspond to localized levels of nitrogen oxide emissions (many other compounds were also visualized) [26]. Similar to Menard with Poland, the designers have used the physical geography to define the surface over which features are presented.

The relationship between 3D surfaces and landscapes is natural. As such, humans enjoy the visual landscape metaphor and seek out known landmarks and features which define it. Landscapes[12] are an organic visual representation for both artificial surfaces (Figure 14) and natural/geographical landscapes (Figure 15). While the landscape surface changes with the underlying data, the user will be able to easily scan and detect the changes. Effective visual "scanning" of surfaces depends on the quality of the graphic design, which ought to resonate with a user's psychophysical landscape. For surfaces, changes in the surface height, the color saturation or texture can be easily visually detected.

The scanning of the artificial surface representing upwards of six dimensions driven by data from a realtime feed is an example of visual data exploration. "Visual data exploration seeks to integrate humans in the data exploration process, applying their perceptual abilities to the large data sets now available. The basic idea is to present the data in some visual form, allowing data analysts to gain insight into it and draw conclusions, as well as interact with it. The visual representation of the data reduces the cognitive work needed to perform certain tasks." [13]. Keim outlines the four theorems for landscape perception:

- "Theorem 1: People seek prospect and refuge as a basic framework for landscape visualization" – A visual baseline is learned by humans through learning the "nominal" landscape.
- "Theorem 2: A landscape is seen to have character through discovery of the details." –Level of Detail (LOD) is defined through links to data dependencies in addition to finer grain views of surface.

- “Theorem 3: Landscapes are viewed as pictorial compositions” – Our first five dimensions are devoted to the landscape surface itself, only when we address the sixth and higher dimensions do we address pictorial compositions.
- “Theorem 4: Visual images of landscape contribute to geographical awareness through cognitive mapping” – When a consistent mapping is defined between the surface features and the visualization domain, then metaphorical geographic awareness will result. The key is to keep the mapping consistent so that humans can learn it and seek refuge in it.

Implementation

We use the Java 3D API [20] to implement a model-viewer-controller (MVC) system for 3D surface creation and navigation [14]. We have chosen Java 3D because it is the only major, openly available scene graph system. It allows us to dynamically change the models in response to changes in the data and user requirements. In addition it can compile the scene graph into either DirectX or OpenGL for compiled execution.

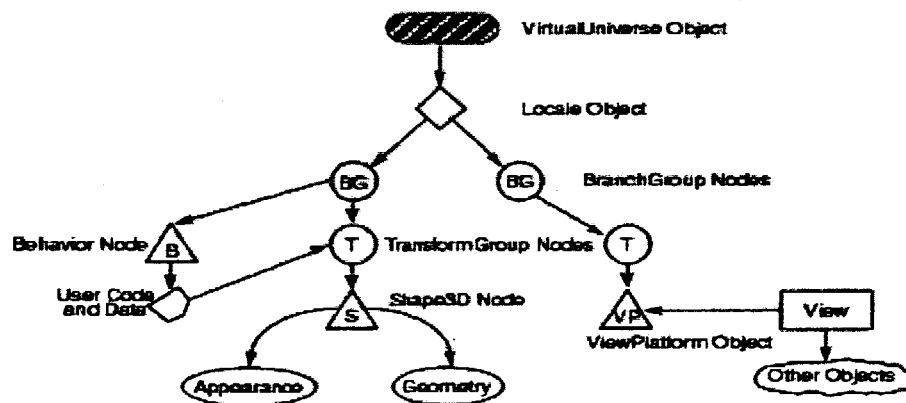


Figure 16 Java3D Scene Graph Model [20]. User must specify, View, Model(Shape3D) and control (Behavior Node).

The top of the scene graph (Figure 16) defines a virtual universe which is made up of a local object which contains the set of branch nodes required of the models and the view. Each branch group consists of a transform group which contains a Shape3D node.

```

<virtual_universe> ::= <locale_object>
<local_object> ::= <branch_group> +
<branch_group> ::= <transform_group>
<transform_group> ::= <shape3D>

```

Shape3D nodes are made up of a geometry and appearance. For our purposes, an appearance is made up of a material, texture, coloring, transparency and rendering. The geometry is defined as a geometry array/indexed face-set made up of a dictionary of coordinates. The face sets are defined for point, line, triangle and quad arrays (Figure 17).


```

<shape3D node> ::= <geometry_object>+ <appearance>
<appearance> ::= [material][texture][coloring][transparency][rendering]
<geometry_object> ::= <geometry_array>
<geometry_array> ::= <coordinate>+
<geometry_array> ::= (point_array | line_array | triangle_array|quad_array)

```

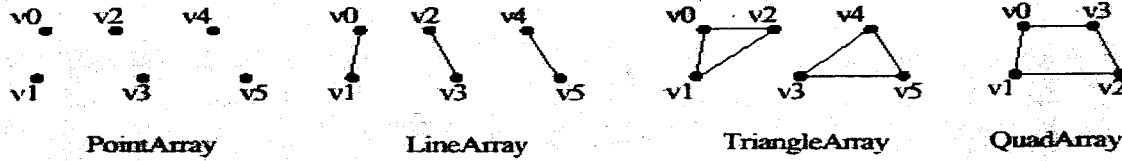


Figure 17 Java3D Geometry Array Types[1]. We utilize both quad and triangle array.

The surface element displays are defined using quad_arrays. Quad arrays are natural for parameterized strip charts because the data for each slice is regular. We will also use triangle arrays for efficiency and minimality. In Figure 17 we illustrate the different types of geometry array that can be created from the point arrays.

The Shape3D constructs ensure that each vertex has a position, color and texture. To create the surfaces we must interpolate the values at each vertex to calculate a position, color and texture for all intermediate points which make up the surface. In Figure 18 we illustrate this challenge by defining the bi-linear interpolation methods for constructing the color and texture of point P_n . This will allow us to define areas and surfaces between the points. Height is taken care of by intersection of quad or triangles with lines while the color and texture of P_n are computed as a function of the four surrounding points:

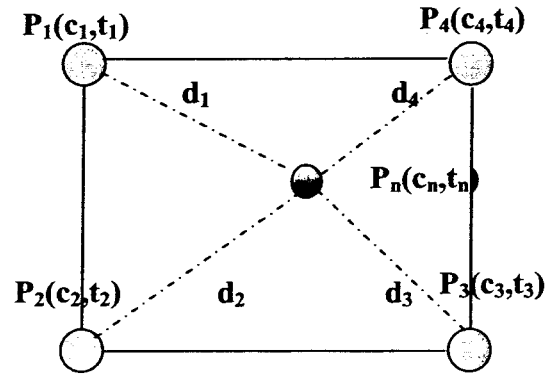


Figure 18. The color and texture of point P_n inside of a quad array, defined using bilinear interpolation function.

$$P_n(c_n, t_n) = \sum_{i=1}^4 f_i(P_i(c_i, t_i))$$

Each coordinate is defined as a tuple and a set of dependencies. Each 3D tuple has attributes for its (x, y, z, color, texture and normal) position. In the program management domain, $\langle x, y, z \rangle ::= \langle \text{time}, \text{plan}, \text{TRL} \rangle$, while in the ISS spacecraft management domain, $\langle x, y, z \rangle ::= \langle \text{time}, \text{processor}, \text{frame count} \rangle$. Color is defined as a range between [0..1], and a texture array (u,v) is defined for each texture attribute (e.g, roughness, shininess, reflection):

```

<coordinate> ::= <tuple> <doc_dependency>*
<tuple> ::= <x> <y> <z> [<color>][<texture>][<normal>]
<x,y,z>_TRL ::= <time, plan, TRL> ;;for TRL domain
<x,y,z> ::= <time, processor, frame count> ;;for frame count domain

```

The dependencies between 3D tuples and source documents are defined by a grammar which maps 3D surface features to source documents. The dependencies are used to query a document management system such as Netmark[22] for access to source documents, or a set of web pages (e.g. ISS ECWA events[29]). An additional paper entitled "Realtime Knowledge Management (RKM) – from an International Space Station (ISS) Point of View" [25] addresses the issues of defining the dependency grammar.:

`<doc_dependency> ::= <document identifier> <page>+`

Conclusion

"A 3D surface is worth ten thousand data points"

The use of 3D information visualization methods will help NASA program managers and spacecraft mission controllers gain insight into the complex systems they monitor and control. These methods will also help in advanced mission control concepts. One such concept is Gemini at NASA Johnson Space Center, whereby just a few people, who are responsible for all mission control console positions at once, monitor the whole of ISS[4]. Recent and past reports analyzing NASA failures highlight the fact that **NASA did know about the root causes of its failures**, but that the relevant information did not flow to those people in position of authority to use it. Our task: how to present an abstract view to filter unnecessary details and at the same time allow for access to relevant details? These two criteria work against each other.

Our approach is to develop 3D surfaces from ordered homogenous strip charts. We note that both program management and spacecraft management domains measure performance by monitoring parameters over time in the form of strip charts. The timescale of the strip charts can vary through orders of magnitude, whether it is monthly status reports, or once a second telemetry downlink. Strip charts are very important because they unite symbolic and numeric reasoning systems by time-stamping each type. The 3D surface defines an abstract view over the low-level data points in the strip charts. This abstract view can be rapidly searched by the viewer due in part to the fact that 3D surfaces devote a greater percentage of the screen area for parameter differences than overlaid 2D strip charts. A drawback of the 3D surfaces to note is that the strip charts for the surface must be ordered; if no order can be found the value of the approach may be questionable. Still, the 3D surface provides opportunities not available for overlaid strip charts by enabling the mapping of additional domain parameters to 3D surface color and texture.

We utilize Java 3D to model the system using a model-viewer-controller paradigm. We have augmented the standard scene graph models with dependencies. The underlying dependency mechanism will relate scene graph primitives to document and source material indices stored in systems such as Netmark. In related work in the ECS Iron Bird Workshop we explore more fully the grammar for document dependencies [25]. In future work, we seek to extend the number of dimensions we can model and still maintain cognitive coherence for the information consumer. In addition we will develop methods to play realtime telemetry and diagnoses through the visualizations.

Acknowledgements

This work funded by the Intelligent Systems (IS) and Engineering for Complex Systems (ECS) programs. We would also like to thank Daryl Fletcher (SAIC) of the NASA Ames Code IC Mobile Systems Lab for help in acquiring the ISS computer frame count data, Linda Timucin (RIACS) for her leadership on the ECS Infoviz project and Beth Minneci (ASANI) of the NASA Ames Code IC Outreach Group for technical editing.

References

- [1] Mankins, J. C. Technology Readiness Levels – A White Paper April 6 1995 NASA HQ – <http://www.advtech.jsc.nasa.gov/downloads/TRLs.pdf>
- [2] “NASA Integrated Action Team (NIAT) Report” – December 21 2000, <ftp://ftp.hq.nasa.gov/pub/pao/reports/2001/NIAT.pdf>
- [3] “Columbia Accident Investigation (CAIB) Report” http://www.caib.us/news/report/pdf/vol1/full/caib_report_volume1.pdf
- [4] Koerner, S., Mission Operations (MOD) Gemini Operations Concept – Baseline Release 3/03
- [5] International Space Station (ISS) Command and Data Handling Manual CDH TM 21109 – Mission Operations Directorate Space Flight Training Division 2/10/2003
- [6] Gawdiak, Y. NASA Aviation Safety AvSP Program 2000
- [7] Gawdiak, Y. NASA AVSP Monthly Report September 2000
- [8] Fletcher, D. P., Alena, R. “A Scalable, Out-of-Band Diagnostics Architecture for International Space Station Systems Support”, IEEE Aero 2003
- [9] D684-10056-01K ISS Prime Contractor Software Standards and Procedures Specification 12/00
- [10] Robinson, P., Shirley M. Fletcher, D., Alena, R., Duncavage, D., Lee, C. “Applying Model-Based Reasoning to the FDIR of the Command & Data Handling Subsystem of the International Space Station,” iSAIRAS 03 2003
- [11] Gershon, N., Page, W. “Discovering Visual Metaphors” Special Issue of Communications of the ACM – Visualize Everything August 2001 – Volume 44, Number 6
- [12] Jakle, J. A. “The Visual Elements of Landscape” The University of Massachusetts Press Amherst 1987
- [13] Keim, D. “Visual Exploration of Large Data Sets” Special Issue of Communications of the ACM – Visualize Everything August 2001 – Volume 44, Number 6
- [14] Barrileaux, J. “3D User Interfaces with Java 3D – A Guide to computer-human interaction in three dimensions” Manning Publications 2001
- [15] Tufte, E. The Visual Display of Quantitative Information Graphics Press Cheshire Connecticut 1983
- [16] Brand, Stewart “The Media Lab: Inventing the Future at MIT. Andrew Lippman on Interactivity”, New York: Viking, 1987.
- [17] Tufte, E. Visual Explanations Graphics Press Cheshire Connecticut 1997
- [18] Tufte, E. Envisioning Information Graphics Press Cheshire Connecticut 1990
- [19] Introduction to Java 3D (1) Class Notes: 240-302, Computer Engineering Lab IV (Software) Dr. Andrew Davison HTUdandrew@ratree.psu.ac.th UTH